UDACITY

PROJECT

Predicting Boston Housing Prices

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Data Exploration

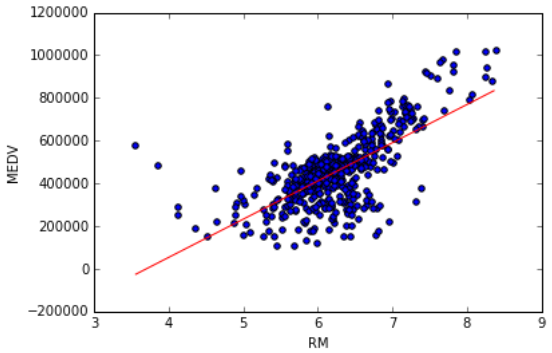
All requested statistics for the Boston Housing dataset are accurately calculated. Student correctly leverages NumPy functionality to obtain these results.

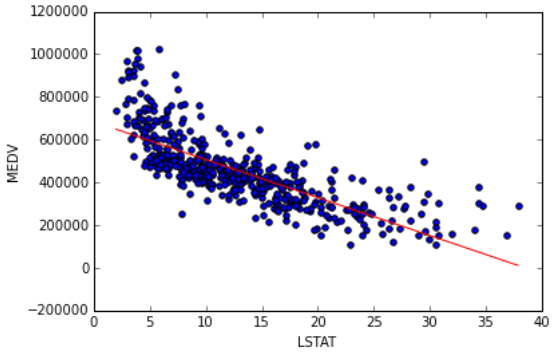
Great job here! Looking at some basic descriptive statistics before we dive deeper into our data can help us steer clear of assumptions that might not necessarily apply (e.g. assuming the data is normally distributed when in fact it is not). It also help us get a feeling for what a sensible prediction value might be.

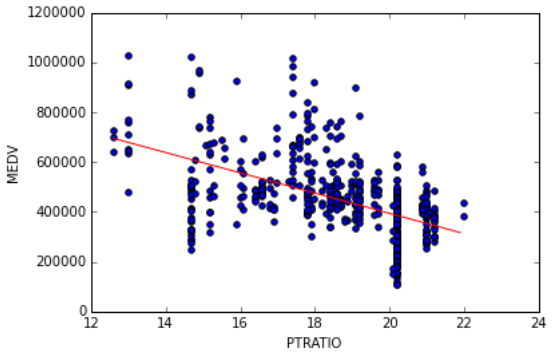
Student correctly justifies how each feature correlates with an increase or decrease in the target variable.

We can always check our hypotheses by plotting the corresponding features against one another along with the line of best fit:

```
for feature in ["RM","LSTAT","PTRATIO"]:

    #Scatter plot.
    ax = data.plot(kind="scatter", x=feature,y="MEDV")

    #Line of best fit (polynomial of degree 1).
    w1, w0 = np.polyfit(data[feature],data["MEDV"],deg=1)

    #Plot line of best fit.
    rng = np.arange(np.min(data[feature]),np.max(data[feature]),0.1)
    ax.plot(rng,[w0+w1*x for x in rng],color="red")
```







Developing a Model

Student correctly identifies whether the hypothetical model successfully captures the variation of the target variable based on the model's R^2 score.
The performance metric is correctly implemented in code.

> From the performance metric output, the R2 score of the model is close to 1.0, which indicates that its predictions are rather satisfactory.

Correct but notice that the coefficient of determination (R^2) estimates the proportion of the target variable's variance that is captured by the model.

Student provides a valid reason for why a dataset is split into training and testing subsets for a model. Training and testing split is correctly implemented in code.

> It's essential to keep some data for testing, which should be kept unseen by the model during the training period, to verify the real performance of predicting after training.

Correct, an independent testing set helps us produce a better evaluation of how well the model is likely to perform when faced with unseen data.

Analyzing Model Performance

Student correctly identifies the trend of both the training and testing curves from the graph as more training points are added. Discussion is made as to whether additional training points would benefit the model.

> At first, the training scrore starts from 1.0, and the curve has a negative slope(score decreases), as more training points are added
>
> While the testing score obviously starts from zero, and increases sharply when num_of_training_points < 50, then it increases slower when 50 < num_of_training_points < 150, and stablishes at about 0.8

Correct, the training and testing scores decrease and increase respectively before converging.

> But that's it, the R2 score of the training curve and testing curve tends to converge as more and more data added after reaching 250, therefore the slope of two curves doesn't change much.

Correct, once the training and testing scores both converge it is clear that the model won't profit from training on more data.

Student correctly identifies whether the model at a max depth of 1 and a max depth of 10 suffer from either high bias or high variance, with justification using the complexity curves graph.

> When the model is trained with a maximum depth of 1, the model suffers from high bias problem. Because both the training score and validation score are low, which indicates that the model with underfits the data.

Correct, relatively low training and testing scores suggest that the model suffers from high bias.

> When the model is trained with a maximum depth of 10, the model suffers from high variance problem. Because while the training score is so high, the valiadation score is much lower

Correct, a training score that is much higher than its corresponding testing score suggests that the model suffers from high variance.

Student picks a best-guess optimal model with reasonable justification using the model complexity graph.

Evaluating Model Performance

Student correctly describes the grid search technique and how it can be applied to a learning algorithm.

> Grid Search is like a paramter sweep which tries exhaustive searching through a manually specified subset of the hyperparameter space.

Correct, grid search exhaustively tries every combination of the provided hyper-parameter values in order to find the best possible model.

Student correctly describes the k-fold cross-validation technique and discusses the benefits of its application when used with grid search when optimizing a model.

> We then calculate the average performance of the models based on the different, independent folds to obtain a performance estimate that is less sensitive(lower-variance estimate) to the subpartioning of the training data compared to the traditional hold-out method.

Correct, the average score provides a more accurate and robust evaluation of the model's performance.

> it helps filtering out some rather optimal candidates for Grid Search to try

Correct, the key here is that if we use only one split to produce the score for a given model the score might be noisier than if we train and test the model multiple times (as in k-fold cross-validation) and we use the average score instead. A more robust score allows grid search in turn to make more accurate comparisons between models.

Student correctly implements the `fit_model` function in code.

Student reports the optimal model and compares this model to the one they chose earlier.

Student reports the predicted selling price for the three clients listed in the provided table. Discussion is made for each of the three predictions as to whether these prices are reasonable given the data and the earlier calculated descriptive statistics.

Tip: We can also compare the predicted prices to the average of the 10 nearest neighbors (neighbors in the sense that their feature values are similar - e.g. similar number of rooms, ptratio and lstat).

```
from sklearn.neighbors import KNeighborsRegressor as knn

reg = knn(n_neighbors=10)
reg.fit(features, prices)

reg.predict(client_data)
```

Student thoroughly discusses whether the model should or should not be used in a real-world setting.

DOWNLOAD PROJECT

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH

Rate this review

Student FAQ